

**UNITED STATES PATENT APPLICATION**

of

**Donald J. Kadyk**

**Neil S. Fishman**

**Marc E. Seinfeld**

for

**ACCOUNTING FOR UPDATE NOTIFICATIONS IN  
SYNCHRONIZING DATA THAT MAY BE REPRESENTED  
BY DIFFERENT DATA STRUCTURES**

WORKMAN, NYDEGGER & SEELEY

A PROFESSIONAL CORPORATION

ATTORNEYS AT LAW

1000 EAGLE GATE TOWER

60 EAST SOUTH TEMPLE

SALT LAKE CITY, UTAH 84111

## **BACKGROUND OF THE INVENTION**

### **1. The Field of the Invention**

The present invention relates to synchronizing data. Specifically, the present invention relates to methods, systems, and computer program products for synchronizing data stored at one or more message clients with data stored at a message server, where the clients may receive update notifications and may represent the data using different data structures than the message server uses to represent the same data.

### **2. Background and Related Art**

There are a variety of circumstances where it is desirable to maintain the same data in different locations. Synchronizing data is the process of reflecting in one copy of the data, changes that are made in another. Depending on the needs of a particular application, synchronization may be almost instantaneous or may be relatively sporadic. During the time that elapses between data synchronizations, a copy of data that has not been updated is termed "stale."

One increasingly popular application for maintaining two copies of the same data is the use of personal digital assistants ("PDAs"). PDAs store various types of information, including task, calendar, contact, email, music, map, financial, and sports data. In short, PDAs may store virtually any type of electronic information. The compact size of PDAs allows a user the convenience of having access to a large amount of information without the bulk of a portable computer or paper organizer. In many cases, a host computer provides centralized data storage for a PDA, with the PDA and host computer each storing separate copies of the data.

1 Note that in this description and in the claims, a second device containing a  
2 “copy” of the data on a first device means that the second device represents the current  
3 state of the data insofar as the second device is capable of representing, or chooses to  
4 represent, the data stored on the first device, given the hardware and software capabilities  
5 of the second device. For example, data formats, representations, and supported fields  
6 may vary from one device to another due to corresponding differences in application  
7 software, operating systems, available memory, processor type, etc. One reason that such  
8 differences exist, and are likely to continue, is a lack of standardization. Nevertheless,  
9 for purposes of this description, the second device has a “copy” of the data on the first  
10 device if the second device represents the current state of the data insofar as the second  
11 device is either capable or chooses, considering the hardware and software differences  
12 that may exist between the devices.

13 The host’s centralized storage offers a number of benefits. For example, it allows  
14 several users to share task, calendar, and contact data and the host may act as a primary  
15 location for receiving email. However, the PDA and host do not maintain an open  
16 channel of communication in typical operation. As a result, the data stored at either the  
17 host or PDA becomes stale because changes made in one device cannot be reflected in  
18 the other. Therefore, the host and PDA data must be updated periodically to insure that  
19 each device contains current information.

20 There are two basic types of data updates: demand synchronization and update  
21 notification. A demand synchronization is an explicit request to synchronize data.  
22 Generally, a demand synchronization is characterized by two-way communication  
23 between the synchronizing devices. During a demand synchronization, the devices  
24 exchange any modified data. The synchronization is complete once the devices confirm

1 to each other that they have received all modifications and contain the same data insofar  
2 as the devices are capable of representing, or choose to represent, the data.

3 In contrast, update notifications are one-way communications that simply inform  
4 another device of data changes as they occur. Because they involve one-way  
5 communications, update notifications may be carried out over an unreliable  
6 communication channel, whereas demand synchronizations commonly are performed  
7 over a reliable communication channel. (In general, an unreliable communication channel  
8 does not include a destination acknowledging the receipt of data, whereas a reliable  
9 channel includes an acknowledgement.)

10 Demand synchronization usually includes the following operations. Both the host  
11 and the PDA track changes in their data that have occurred since the previous  
12 synchronization. Then, at the time of synchronization, only the tracked changes are  
13 exchanged between the host and PDA. If the same data has been changed in both devices,  
14 the synchronization recognizes a potential conflict and prompts the user to determine  
15 which change should be retained. In the absence of a conflict, changes are applied  
16 automatically, without prompting the user. Any changes made during the demand  
17 synchronization process are tracked as synchronization changes so that these changes will  
18 not be exchanged during a subsequent demand synchronization. Otherwise, a PDA  
19 receiving a changed phone number from a host would reflect the changed phone number  
20 back to the host during the next synchronization, even though the PDA made no change  
21 to the phone number, other than to make it consistent with the phone number stored at the  
22 host.

23 Resolving conflicts without user intervention requires relatively sophisticated  
24 processing and as a rule the user is prompted for all but the most trivial cases. One reason

1 why it may be difficult to resolve conflicts automatically is that, as mentioned above, a  
2 PDA may represent, format, or store information differently from how the corresponding  
3 information is stored at the host. As a result, even data that is the “same” may contain  
4 different information.

5 One area of recent advancement in PDA technology involves more frequently  
6 updating a PDA through update notification in order to reduce data staleness. Previously,  
7 PDAs had included only cable or infrared links for communicating with a host, requiring  
8 the two devices to be in relatively close proximity for even demand synchronization.  
9 Terrestrial airwave receivers have been added to some newer PDA models in order to  
10 increase the range at which they can receive new or changed data from a host. The  
11 increased range allows for both update notification and demand synchronization to occur  
12 while a PDA is located a substantial distance from a host.

13 Transmitting over substantial distances introduces some uncertainty as to whether  
14 or not a PDA will receive an update notification. For example, the PDA may travel  
15 beyond the transmission area and thus never receive the update notification. Since the  
16 communication links for update notification may be unreliable (in contrast to the reliable  
17 links commonly used in demand synchronization), the host remains ignorant of the  
18 unsuccessful receipt of the update notification. Thus, the host does not know to take any  
19 corrective action such as resending the update notification.

20 After processing an update notification, it may seem as if both devices have  
21 independently made a change since the same change may appear different depending on  
22 how the devices store the data. Thus, during a subsequent demand synchronization, the  
23 apparent conflict is likely to require user intervention to resolve. A user must compare the  
24 PDA and host data and confirm that they in fact represent the same data. Alternatively,

1 demand synchronization may not be able to determine whether or not a particular change  
2 was received and, as a result, send a duplicate of an earlier change. For example, a new  
3 contact may be added once by an update notification and a second time by a demand  
4 synchronization. Resolving duplicate changes also requires user intervention. However,  
5 requiring user intervention substantially erodes the benefits offered by update  
6 notification. It can be frustrating and inordinately time-consuming for a user to make  
7 even a few comparisons or reconcile a few duplicate records.

8 As illustrated by the preceding examples, a need exists for synchronizing  
9 efficiently when update notifications may be received and when synchronizing between  
10 devices that represent the same data using different data structures. Therefore, methods,  
11 systems, and computer program products are desired that allow for the benefits of update  
12 notifications and account for differences in how information is stored, without imposing  
13 substantial burdens on the user.



1 The present invention associates tokens with stored data so as to reconcile  
2 changes that occur in separate “copies” of the same data. As mentioned above, a second  
3 device has a “copy” of the data on a first device if the second device represents the  
4 current state of the data insofar as the second device is able or chooses. This means that  
5 the copy may not be an identical replica of the original data.

6 The tokens serve at least two purposes: (1) they identify specific data, and (2)  
7 they identify a particular version of the specific data. Consider, for example, an  
8 individual data object in a contact database. As changes to the data object occur, a series  
9 of tokens may be generated, one for each version of or update to the data object. While  
10 all of the tokens in the series identify the same data object, each individual token  
11 identifies a different version of the data object. (That is, the data identifying part of each  
12 token is the same, but the data version part of each token is different.) If these changes  
13 are synchronized over an unreliable communication channel, it may appear as if separate  
14 changes have been made to each copy of the data, since different devices may represent  
15 the same data using different data structures. However, by reference to the tokens  
16 identifying the changes in question, determining whether two apparently distinct changes  
17 are in fact the same change is greatly simplified, especially in systems where different  
18 devices represent the same data using different data structures.

19 Initially, a sending device and a receiving device contain the “same” data. In other  
20 words, the receiving device represents the current state of the data insofar as it is able or  
21 chooses. At a subsequent time, the data stored in the sending device is modified in some  
22 way and the sending device creates a token that uniquely identifies the change. The token  
23 is unique to the sending device. Once the token has been created, the sending device  
24 generates a notification that includes both the change and corresponding token. This



1 notification is sent to the receiving device in order to synchronize the data stored at the  
2 receiving device with the data stored at the sending device. (The sent token may be  
3 compressed to reduce its size. Although the compressed token may no longer be unique  
4 to the sending device, it is unique to the receiving device.)

5 Eventually, the receiving device requests demand synchronization to insure that it  
6 has received all changes that have occurred at the sending device. To perform this check,  
7 the receiving device returns a starting token and all the tokens it has received to the  
8 sending device. (If the tokens were compressed, they must be decompressed.) The  
9 sending device then compares the returned tokens with the tokens the sending device  
10 sent. Any tokens after the starting token that have not been returned by the receiving  
11 device indicate that the receiving device did not receive the change associated with the  
12 missing token. In response to a missing token, the sending device resends the change.  
13 Also, any new tokens provided by the receiving device indicate changes to data on the  
14 receiving device that should be delivered to the sending device.

15 Tokens also may be used to divide a change into various portions. For example,  
16 an email may be divided into a header portion, a body portion, and an attachment portion.  
17 Initially, a sending device may send a notification that corresponds to the header portion.  
18 This allows the receiving device to display a representative amount of the message, such  
19 as who originated the message, the subject line, etc., without having to receive the full  
20 message. Then, in response to receiving the token corresponding to the header portion,  
21 the sending device may send the remaining portions of the email message. A missing  
22 header token would result in the sending device sending the full email message.

23 Additional features and advantages of the invention will be set forth in the  
24 description which follows, and in part will be obvious from the description, or may be

1 learned by the practice of the invention. The features and advantages of the invention  
2 may be realized and obtained by means of the instruments and combinations particularly  
3 pointed out in the appended claims. These and other features of the present invention will  
4 become more fully apparent from the following description and appended claims, or may  
5 be learned by the practice of the invention as set forth hereinafter.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

In order that the manner in which the above-recited and other advantages and features of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

Figure 1 illustrates an exemplary system that provides a suitable operating environment for the present invention;

Figure 2 is block diagram showing data structures and communication channels for synchronizing client data with server data;

Figures 3A and 3B depict a server perspective flow chart of a method for synchronizing two copies of the same data; and

Figures 4A and 4B illustrate an example of the synchronization of a client copy of contact information with changes made to a server copy of the same information in accordance with the present invention.

**DETAILED DESCRIPTION OF THE INVENTION**

The invention is described below by using diagrams to illustrate either the structure or processing of embodiments used to implement the systems, methods, and computer program products of the present invention. Using the diagrams in this manner to present the invention should not be construed as limiting of its scope. The present invention extends to methods, systems, and computer program products for synchronizing data stored at one or more message clients with data stored at a message server.

As used in the application, a message server is a device or combination of devices that originates changes to data and a message client is one or more devices that receive the originated changes from the message server. The same device may operate, therefore, as either a message server or a message client, depending on how it interacts with other devices. It should also be recognized that references to copies of the “same” data includes copies containing of only a subset of the data in question. In other words, one copy may include only selected fields from another copy and still be considered to store the “same” data.

Those of skill in the art will recognize that the principles of the present invention allow for efficient demand synchronization when a message client may receive update notifications from a message server, as noted in the following and other considerations. First, update notifications that the message client receives are communicated with demand synchronization requests so that demand synchronization does not duplicate the changes the message client received through update notification. Second, demand synchronization provides a change if an update notification is not received or if an update notification includes only a portion of the change. For example, an update notification

1 may include only certain portions of an email message, such as a subject line, the sender,  
2 etc. Demand synchronization provides other portions that were not included in the update  
3 notification. Third, changes may be identified regardless how a particular device stores  
4 data. In many cases, data formats, representations, and supported fields vary from one  
5 device to another, based on corresponding differences in application software, operating  
6 systems, available memory, processor type, etc.

7       The message server associates a token with changes that are made to the data  
8 stored at the server. Note that the term “change” should be interpreted broadly to include  
9 a modification to existing data as well as the addition of new data. A token identifies both  
10 the data that was changed (e.g., the specific data object that was modified) and the  
11 revision of the data that the change represents. The tokens and changes are combined to  
12 form notifications that are sent to the message clients. Tokens may be compressed to  
13 conserve storage space, but the tokens should be unique to each message client if not  
14 unique to the message server.

15       When clients request synchronization with the message server, the clients return  
16 all the tokens that have been received from the message server. Upon receiving the  
17 returned tokens, the message server compares the tokens returned by the clients with the  
18 tokens that were sent to the clients. The message server interprets a clients’ failure to  
19 return a particular token as an indication that the token and corresponding change were  
20 never received by the client. In response, the message server resends the change to the  
21 client.

22       Tokens also may be used to subdivide a change. Only part of the change is  
23 included within the notification sent to the clients. The message server then interprets a  
24

1 client returning a token as an indication that the client would like to receive the remainder  
2 of the change. If the token is not returned, the server sends the full change.

3       Embodiments within the scope of the present invention include  
4 computer-readable media for carrying or having computer-executable instructions or data  
5 structures stored thereon. Such computer-readable media can be any available media  
6 which can be accessed by a general purpose or special purpose computer. By way of  
7 example, and not limitation, such computer-readable media can comprise physical  
8 storage mediums such as RAM, ROM, EEPROM, CD-ROM or other optical disk storage,  
9 magnetic disk storage or other magnetic storage devices, or any other medium which can  
10 be used to carry or store desired program code means in the form of computer-executable  
11 instructions or data structures and which can be accessed by a general purpose or special  
12 purpose computer. Such a medium may include a wireless carrier signal, for example.  
13 When information is transferred or provided over a network or another communications  
14 connection (either hardwired, wireless or a combination of hardwired and wireless) to a  
15 computer, the computer properly views the connection as a computer-readable medium.  
16 Thus, any such connection is properly termed a computer-readable medium.  
17 Combinations of the above also should be included within the scope of  
18 computer-readable media. Computer-executable instructions comprise, for example,  
19 instructions and data which cause a general purpose computer, special purpose computer,  
20 or special purpose processing device to perform a certain function or group of functions.

21       Figure 1 and the following discussion are intended to provide a brief, general  
22 description of a suitable computing environment in which the invention may be  
23 implemented. Although not required, the invention may be described in the general  
24 context of computer-executable instructions, such as program modules, being executed

1 by computers in network environments. Generally, program modules include routines,  
2 programs, objects, components, data structures, etc. that perform particular tasks or  
3 implement particular abstract data types. Computer-executable instructions, associated  
4 data structures, and program modules represent examples of the program code means for  
5 executing steps of the methods disclosed herein. The particular sequence of such  
6 executable instructions or associated data structures represent examples of corresponding  
7 acts for implementing the functions described in such steps.

8 Those skilled in the art will appreciate that the invention may be practiced in  
9 network computing environments with many types of computer system configurations,  
10 including personal computers, hand-held devices, multi-processor systems,  
11 microprocessor-based or programmable consumer electronics, network PCs,  
12 minicomputers, mainframe computers, and the like. The invention may also be practiced  
13 in distributed computing environments where tasks are performed by local and remote  
14 processing devices that are linked (either by hardwired links, wireless links, or by a  
15 combination of hardwired and wireless links) through a communications network. In a  
16 distributed computing environment, program modules may be located in both local and  
17 remote memory storage devices.

18 Figure 1 illustrates a conventional computer 120 that includes components and  
19 data processing capabilities that may be used to implement embodiments of the  
20 invention. Computer 120 is a general purpose computing device that includes a  
21 processing unit 121, a system memory 122, and a system bus 123 that couples various  
22 system components including the system memory 122 to the processing unit 121.  
23 Processing unit 121 executing computer-executable instructions is one example of  
24 processing means for performing steps and acts according to the present invention. The

1 system bus 123 may be any of several types of bus structures including a memory bus or  
2 memory controller, a peripheral bus, and a local bus using any of a variety of bus  
3 architectures. The system memory includes read only memory (ROM) 124 and random  
4 access memory (RAM) 125. A basic input/output system (BIOS) 126, containing the  
5 basic routines that help transfer information between elements within the computer 120,  
6 such as during start-up, may be stored in ROM 124.

7 The computer 120 may also include a magnetic hard disk drive 127 for reading  
8 from and writing to a magnetic hard disk 139, a magnetic disk drive 128 for reading from  
9 or writing to a removable magnetic disk 129, and an optical disk drive 130 for reading  
10 from or writing to removable optical disk 131 such as a CD-ROM or other optical media.  
11 The magnetic hard disk drive 127, magnetic disk drive 128, and optical disk drive 130 are  
12 connected to the system bus 123 by a hard disk drive interface 132, a magnetic disk  
13 drive-interface 133, and an optical drive interface 134, respectively. The drives and their  
14 associated computer-readable media provide nonvolatile storage of computer-executable  
15 instructions, data structures, program modules and other data for the computer 120.  
16 Although the exemplary environment described herein employs a magnetic hard disk  
17 139, a removable magnetic disk 129 and a removable optical disk 131, other types of  
18 computer readable media for storing data can be used, including magnetic cassettes, flash  
19 memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs, and the like.

20 Program code means comprising one or more program modules may be stored on  
21 the hard disk 139, magnetic disk 129, optical disk 131, ROM 124 or RAM 125, including  
22 an operating system 135, one or more application programs 136, other program modules  
23 137, and program data 138. A user may enter commands and information into the  
24 computer 120 through keyboard 140, pointing device 142, or other input devices (not



1 shown), such as a microphone, joystick, game pad, satellite dish, scanner, or the like.  
2 These and other input devices are often connected to the processing unit 121 through a  
3 serial port interface 146 coupled to system bus 123. Alternatively, the input devices may  
4 be connected by other interfaces, such as a parallel port, a game port or a universal serial  
5 bus (USB). A monitor 147 or another display device is also connected to system bus 123  
6 via an interface, such as video adapter 148. In addition to the monitor, personal  
7 computers typically include other peripheral output devices (not shown), such as speakers  
8 and printers.

9 The computer 120 may operate in a networked environment using logical  
10 connections to one or more remote computers, such as remote computers 149a and 149b.  
11 Remote computers 149a and 149b may each be another personal computer, a server, a  
12 router, a network PC, a peer device or other common network node, and typically  
13 includes many or all of the elements described above relative to the computer 120,  
14 although only memory storage devices 150a and 150b and their associated application  
15 programs 136a and 136b have been illustrated in Figure 1. The logical connections  
16 depicted in Figure 1 include a local area network (LAN) 151 and a wide area network  
17 (WAN) 152 that are presented here by way of example and not limitation. Such  
18 networking environments are commonplace in office-wide or enterprise-wide computer  
19 networks, intranets and the Internet.

20 When used in a LAN networking environment, the computer 120 is connected to  
21 the local network 151 through a network interface or adapter 153. When used in a WAN  
22 networking environment, the computer 120 may include, for example, a modem 154 or a  
23 wireless link. The modem 154, which may be internal or external, is connected to the  
24 system bus 123 via the serial port interface 146. In a networked environment, program

modules depicted relative to the computer 120, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means for establishing communications over wide area network 152 may be used.

Figure 2 is block diagram showing data structures and communication channels for synchronizing storage 268 of client 260 with storage 218 of server 210. Enlarged portion 220 of storage 218 shows that storage 218 includes data type 230, data type 240, and data type 250. Data types are generic representations of information that may be stored in storage 218. For example, data types may correspond to general types of information such as task, contact, calendar, email, music, financial, and sports data, or data types may correspond to separate fields within data objects of a database. Data 232, data 242, and data 252 are provided to indicate that data types may be further subdivided. If data type 230 were task data, then data 232 might correspond to individual task objects. The present invention does not impose any particular requirements on what data types 230, 240, and 250 or data 232, data 242, and data 252 represent, the data and data types merely indicate that many types of information may be present at storage 218 in a wide variety of formats.

Client 260 and server 210 communicate over communication channels 216 and 266. Communication channel 216 includes an arrow that only points to client 260 as an indication of one-way communication from server 210 to client 260. In contrast, communication channel 266 includes arrows at both ends to indicate two-way communication between server 210 and client 260. One embodiment of the present invention implements communication channel 216 as an unreliable communication channel and communication channel 266 as a reliable communication channel.

Initially, storage 218 of server 210 and storage 268 of client 260 store identical copies of the same data. Recall that for purposes of this description and in the claims, a first device has a “copy” of the data stored at a second device if the first device represents the current data insofar as the first device is able or chooses. For example, a Personal Digital Assistant (PDA) may have limited memory and processing capability or may represent and format data differently based on the PDA’s application software or operating system. Therefore, the PDA may store an abbreviated copy of data stored on a host computer in which memory and processing power is not so much a concern. Nevertheless, the PDA would store an identical copy of the data stored at the host computer if the PDA represents the current state of the data insofar as the PDA is capable or chooses. For example, a PDA may list a contact data object as including a name and a phone number while the host computer may list this and numerous other data fields for the same contact. So long as the name and phone number are current at the PDA, the PDA has an identical copy of the data.

It should be noted that enlarged portion 280 of storage 268 shows only data 282. This indicates that storage 268 may store only a subset of the data at storage 218. For simplicity, assume that data 282 corresponds to data 242 of storage 218. (The choice is arbitrary. Data 282 could correspond to data 232, data 242, and/or data 252. Furthermore, the present invention does not limit data 282 as corresponding to data located at a single server.) As changes are made to data 242, tokens 244 are created to identify the changes. The server 210 maintains an association between tokens 244 and data 242, such that when given a token 244, the server is able to identify the data 242 that corresponds to the token. Each of tokens 244 corresponds to a specific part of data 242.

1 To synchronize data 282 with data 242, server 210 creates update notifications.  
2 Each notification contains at least one of the changes and at least one of the  
3 corresponding tokens. As changes occur, the notifications are sent to client 260 over  
4 communication channel 216. In one embodiment, communication channel 216 is a  
5 wireless communication channel and notifications are transmitted to client 260. Client  
6 260 stores the notifications it receives as notifications 290. Notifications 290 include  
7 changes 292 and tokens 294. In one embodiment, tokens 294 are compressed versions of  
8 tokens 244. Tokens 294 are unique to client 260 and tokens 244 are unique to server 210  
9 (as are tokens 234 and tokens 254). Changes 292 correspond to the changes that are made  
10 to data 242, however it is possible that client 260 will not receive all notifications sent by  
11 server 210 and therefore changes 292 may be only a subset of those changes made to data  
12 242.

13 Client 260 also stores collection 270. Collection 270 represents the state of the  
14 copy of data stored at client 260 during a previous synchronization when data 282  
15 reflected all of the changes that had been made to data 242. Along with collection 270, a  
16 token may be stored to indicate the starting point for the next synchronization. Unlike the  
17 association between tokens 244 and data 242 at server 210, however, the tokens 274 at  
18 client 260 are not directly associated with data 282. The reason for this is that collection  
19 270 represents an earlier state of the data, whereas data 282 is the current state of the  
20 data. In one embodiment of the present invention, collection 270 is compressed.

21 At some point in time, client 260 issues a synchronization request to server 210  
22 through communication channel 266. Server 210 receives tokens 294, a starting point,  
23 and collection 270 with tokens 274 back from client 260. The tokens 294 and tokens 274  
24 received back from client 260 are then compared to the tokens 244 that server 210 had

1 sent to client 260. Server 210 interprets any of tokens 244 after the starting point that do  
2 not have matching tokens 294 or tokens 274 as an indication that client 260 did not  
3 receive the changes corresponding to the missing tokens.

4 In response, server 210 sends a new collection (not shown) and a list of tokens  
5 (not shown) identifying the changes that client 260 is missing. Client 260 then uses the  
6 list to request the missing changes from server 210. The missing changes are sent in the  
7 form of notifications. Once it has received the new collection and all of the missing  
8 changes, client 260 discards notifications 290 because the new collection contains all of  
9 the tokens necessary to represent the current state of data 282. Communication channel  
10 266 is closed after client 260 has received its requested synchronization and server 210  
11 starts the process over again by sending subsequent notifications to client 260 through  
12 communication channel 216.

13 Alternatively, if client 260 is unable to retrieve all of the missing changes,  
14 collection 270 and notifications 290 (including any notifications for missing changes that  
15 were received) are retained and the new collection is discarded. At the next  
16 synchronization opportunity, client 260 passes back collection 270 with tokens 274 and  
17 tokens 294 (including missing notifications and new notifications subsequently received  
18 from server 210). Server 210 performs the token comparison and the synchronization  
19 proceeds as described above. As a result, interrupting synchronization presents only a  
20 minor inconvenience.

21 It should be emphasized that the foregoing is only one example of how  
22 synchronization may occur. The present invention, as described in the appended claims,  
23 allows for many variations and therefore, this example should not be construed as  
24 imposing any additional limitations.

1 Methods of the present invention from the perspective of the server will be  
2 described next with reference to the flow chart depicted in Figures 3A and 3B.  
3 Synchronization begins with the step of a server computer providing notifications (320).  
4 The step of providing notification comprises the acts of making a change (322) to the  
5 data stored at the server computer, generating a token (324) that corresponds to the  
6 change made, compressing the token (326) to reduce its size, and sending a notification  
7 (328) to the client computer.

8 Ordinarily, tokens will be unique to the server that generates them. However, in  
9 some circumstances it may be desirable to reduce a token's size. For example, wireless  
10 communication protocols may limit the amount of data sent in individual packets to  
11 approximately one hundred bytes. Large tokens needed to represent uniquely all changes  
12 made at the server, may require an inordinate proportion of the wireless payload. One  
13 solution is to compress the tokens so that they are unique to individual clients rather than  
14 unique to the server and thereby conserve bandwidth. Usually, only tokens sent to the  
15 client are compressed, whereas any tokens retained by the server to identify the version  
16 of data that the server stores remain uncompressed and unique to the server.

17 The step of determining missing notifications 340 follows. Decision block 342  
18 performs the act of indicating whether the server has received a synchronization request  
19 from the client. If so, the server performs the act of receiving tokens (344) back from the  
20 client and the act of comparing the tokens received with the tokens sent to the client  
21 (346). In the absence of a synchronization request at decision block 342, processing  
22 returns to the beginning of step 320, providing notifications.

23 Turning next to Figure 3B, the acts for providing a collection 330 are shown. A  
24 collection contains the tokens necessary to represent what the state of the client data

1 should be after synchronizing has been completed. The collection is generated (334) at  
2 the server and sent to the client (336). The server also sends a list of tokens identifying  
3 the changes that client is missing based on a starting point provided by the client (not  
4 shown). Using this list, the client requests changes that it has not received.

5 After completing the step of providing a collection 330, the server begins the step  
6 of providing missing notifications 350 so that the client will have any changes that were  
7 not received. Providing missing notifications 350 comprises the acts of sending the  
8 missing notification 354 for each missing token identified (352). Note that step 350  
9 allows for providing the change, token, and/or notification, etc. for a missing token.

10 The step of providing a second portion 360 relates to changes that the server  
11 divides into separate parts. For example, an email message may be divided into header,  
12 body, and attachment parts. Associating a change and token with the header allow for  
13 sending only a part of the message to the client. This approach may be attractive where  
14 the communication channel between the server and client has a relatively low bandwidth  
15 or is unreliable. By sending only the header, the client may display some portion of the  
16 email to a user as an indication that the server has received the complete email. Providing  
17 second portion 360 performs the acts of determining if a token identifies a message  
18 portion (decision block 362) and sending any remaining message portion (364) to the  
19 client.

20 Figures 4A and 4B illustrate synchronizing a client copy 450 of contact  
21 information with changes made to a server copy 410 of the same information using  
22 notifications 430. Initially, as shown in Figure 4A, server data 412 and client data 452  
23 contain the same data. Server data 414 reflects changing the contact's first name from  
24 "John" to "Jon." Token 1 is generated to identify the change. Notification 434 includes

1 the name change and corresponding token. Upon receiving notification 434, client data  
2 454 reflects the name change and stores the token. Arrows 424 and 444 indicate that  
3 notification 434 is communicated in one direction, possibly over an unreliable  
4 communication channel.

5 A fax number and its change identifier, Token 2, have been added to server data  
6 415. Notification 435 contains the new fax number and associated token. As above,  
7 communication between the server and client occurs in one direction (arrow 425). In this  
8 instance, however, client data 455 does not receive notification 435. Therefore, neither  
9 the new fax number nor its corresponding token are reflected in client data 455.

10 Server data 416 show a final change. "Jr." is added to the name field and Token 3  
11 has been generated to identify the change. Notification 436 includes the name change and  
12 token. After receiving notification 436, client data 456 reflects the name change and  
13 stores Token 3. Here again, arrows 426 and 446 indicate that communication occurs from  
14 the server to the client without acknowledgement or confirmation.

15 Turning now to Figure 4B, the server has received a synchronization request from  
16 the client. Client data 457 returns to server data 417 all of the tokens received by the  
17 client, including a token for use as a start point that is based on the previous  
18 synchronization. Reference 437 includes Token 1 and Token 3. Token 1 is the starting  
19 point for determining which tokens have been received. Note that Token 2 is not returned  
20 because the client never received notification 435. Reference 437 appears as a dashed box  
21 to indicate that it is a return of received tokens rather than a notification. Arrows 427 and  
22 447 show that returning tokens uses two-way communication between client and server,  
23 possibly implemented as a reliable communication channel. It should be noted that the  
24 present invention does not necessarily impose any particular requirement on how the



1 tokens are returned to the server (e.g., as a collection, notification, or as individual  
2 tokens).

3 Beginning with the starting token, returned tokens are compared with the sent  
4 tokens to identify any tokens that may be missing. Other than identifying a starting point  
5 for the token comparison, the present invention does not impose any particular  
6 requirements on the starting token. Server data 418 shows that Token 2 was not received  
7 back from the client. In one embodiment, the server provides a list of missing tokens  
8 back to the client so that the client may request the changes that it has not received. In  
9 response to a request for Token 2, notification 438 is generated to resend the missing  
10 change and associated token. Alternatively, the server could simply send notification 438  
11 (or the corresponding change itself) rather than sending a list of missing tokens and  
12 waiting for the client to request the corresponding changes.

13 It is possible to send the missing change to the client as notification 438 rather  
14 than only the change itself, but the invention does not necessarily impose any particular  
15 structure for resending changes to the client. Nevertheless, by sending notification 438, if  
16 the resending process is interrupted, it can be resumed without any special processing.  
17 The client simply can proceed at a later time by returning all of the tokens it has received.  
18 Any notifications received during the prior resending provide a corresponding token.  
19 Those notifications not received due to the interruption do not provide a token. The  
20 tokens that are still missing due to the interruption are identified and the corresponding  
21 notifications are sent to the client.

22 As part of synchronizing, the server sends to the client a collection of tokens  
23 representing the current state of the data stored by client. After receiving the collection,  
24 the client discards the notifications received prior to the synchronization and begins